Intensional Approaches for Symbolic Methods

Olga Kushnarenko, Sophie Pinchinat

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Intensional Approaches for Symbolic Methods

Olga Kushnarenko and Sophie Pinchinat

No 3448

Juillet 1998

THÈME 1

Rapport de recherche

ISSN 0249-6399

Intensional Approaches for Symbolic Methods

Olga Kushnarenko and Sophie Pinchinat*

Thème 1 — Réseaux et systèmes
Projet EP-ATR

Rapport de recherche n° 3448 — Juillet 1998 — 18 pages

Abstract: We present a behavioral model for discrete event systems based on an intensional formalism, as a possible approach within the broader trend towards rich symbolic representations in verification. We define *Intensional Labeled Transition Systems* with associated combinators of parallel composition and event hiding, and we propose *symbolic bisimulation* to handle strong bisimulation intensionally. Further on, we explain how the methodology has been developed for the synchronous language SIGNAL, via the verification tool SIGALI.

Key-words: Intensional transition systems, polynomials, (symbolic) bisimulation, synchronous languages, equivalence checking

(Résumé : tsvp)

Work supported by the Esprit SYRF Project 22703.

* email: `{okouchna, pinchina}@irisa.fr`

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex (France)
Téléphone : 02 99 84 71 00 - International : +33 2 99 84 71 00
Télécopie : 02 99 84 71 71 - International : +33 2 99 84 71 71

Approches intentionnelles pour les méthodes symboliques

Résumé : Nous présentons un modèle de comportement destiné aux systèmes d'événements discrets qui se repose sur un formalisme intentionnel. C'est une approche possible dans le cadre de travaux dont le but est d'enrichir et/ou de simplifier des manipulations des représentations symboliques dans le domaine de la vérification. Nous définissons des *système de transitions étiquetés intentionnels* ainsi que des opérateurs de composition parallèle et de masquage d'événements, et nous proposons une bisimulation symbolique afin de manipuler de la bisimulation forte intentionnellement. Nous expliquons comment cette méthodologie a été développée dans le cadre du langage synchrone SIGNAL par moyen de l'outil de vérification SIGALI.

Mots-clé : systèmes de transitions intentionnels, polynomiaux, bisimulation symbolique, langages synchrones, vérification d'équivalences

1 Introduction

Dynamic systems are systems that evolve according to their environment. In general, an evolution of the system, in a given state, depends on an input event (some information given by the environment); this evolution leads to some instantaneous output event and to state changes.

Synchronous languages have been designed to ease the programmer's task when dealing with such systems; they provide some primitives for concurrency and communication. They can be of different kinds. The most popular ones that have been designed in France are: ESTEREL [BC84] an imperative language, LUSTRE [Pla88] and SIGNAL [BLJ91] based on declarative approach. These languages naturally bear a semantics in terms of discrete event systems, and their control part concerns boolean valued signals. The synchronous features allow one to express synchronization constraints between the different (output and internal) events of the system and the input events of its environment. Hence, any operational semantics of such systems leads to automata labeled combinations of atomic events.

The automata semantics can then be used as a basis for the verification of SIGNAL programs. For classic temporal logics specification verifications, the tool SIGALI [DLB97] was developed; this tool is based on an intensional representation of the automata. Whereas SIGNAL programs equivalence checking was made extensionally by feeding other verification tools, e.g. such as ALDEBARAN [Fer84] or FCTOOLS [BRRD96], with the extensional description of the automata. Obviously, the size of the generated transition systems limits the extensional methods.
In this paper, we propose an intensional formalism based on the algebraic theory of polynomials for bisimulation checking which perfectly fits the spirit of the tool SIGALI. The "polynomial language" provides the programmer with an intermediate language to describe symbolic algorithms, in an intensional way, without bothering with the underlying implementation.

In our modelization approach, instead of considering extensionally all possible events for a given state change, we develop a formalism where actions of the automata are polynomials, these automata will be called *intensionally Labeled Transition Systems* (or *iLTS* for short). These polynomials are based on several variables (one for each atomic event) with coefficients in $\mathbf{Z}_3$, according to the following encoding: an atomic boolean event can either be *absent*, then encoded by 0, or *present* and equal to *true*, encoded by 1, or *present* and equal to *false*, encoded by -1. The solutions of a polynomial are composed events. iLTS naturally possess an interpretation in terms of classical labeled transition systems, but they permit to avoid the transition enumeration one would get by describing extensionally each event and transition. Moreover, the algebraic theory of polynomials offers simple definitions for *parallel composition* and *event hiding*, both combinators widely used to design complex systems.

The paper is organized as follows: Sections 2.1 and 2.2 introduce the intensional models and the combinators. Further on, Section 2.3, we propose a behavioral equivalence, called *symbolic bisimulation* over iLTS with good properties; it has the congruence property w.r.t. the combinators (see Theorem 6). This definition enables one to handle classic strong bisimulation in an intensional style (see Theorem 8). Then, in order to proceed to symbolic verification, Section 2.4 introduces a still more intensional semantics for systems: polynomial formalism is extended to describe the whole system, that is, all its legal transitions. The resulting models are called *Intensional Labeled Transition Systems* (ILTS). Section 3 explains how the developed theory is currently applied to the language SIGNAL: the options of the compiler plugged with the basic functions of the verification tool SIGALI [DLB97] allow us to perform polynomial handling for bisimulation computation. For lack of space, we refer to [KP98] for the proofs details.

2 Intensional Labeled Transition Systems

This section introduces *intensionally labeled transition systems*, *parallel composition* and *event hiding*, as well as the *symbolic bisimulation* behavioral equivalence. Then *intensional labeled transition systems* are proposed as a more intensional description for labeled transition systems, and an algorithm for the symbolic bisimulation computation is given.

In the following, we write $\mathbf{Z}_3$ for the finite field $\{-1, 0, 1\}$ in which $x^3 = x$ and $3x = 0$ for all $x \in \mathbf{Z}_3$. Let $\bar{Z}$ be a finite set of m distinct variables $Z_1, ..., Z_m$. We denote by $\mathbf{Z}_3[\bar{Z}]$ (or $\mathbf{Z}_3[Z_1, ..., Z_m]$) the set of polynomials over variables $Z_1, ..., Z_m$ which coefficients range over $\mathbf{Z}_3$ with typical elements $P(\bar{Z})$ (or P for short), $P_1(\bar{Z}), \ldots$. We recall that $(\mathbf{Z}_3[\bar{Z}], +, *)$ is a ring.

2.1 Intensionally labeled transition systems

Definition 1. (Intensionally Labeled Transition Systems (iLTS)) An *m-dimensional intensionally Labeled Transition System (or m-iLTS)* is a structure
$T = (Q, \bar{Z}, \rightarrow)$, where

- Q is set of states,
- $\bar{Z}$ is a set of m variables $Z_1, ..., Z_m$, and
- $\rightarrow \subseteq Q \times \mathbf{Z}_3[\bar{Z}] \times Q$. Each transition is labeled by a polynomial over the set $\bar{Z}$.

We write $q \overset{P(\bar{Z})}{\rightarrow} q'$ (or simply $q \overset{P}{\rightarrow} q'$), instead of $(q, P(\bar{Z}), q') \in \rightarrow$.
Given a polynomial $P(\bar{Z}) \in \mathbf{Z}_3[\bar{Z}]$, we associate its set of solutions $Sol(P) \subseteq \mathbf{Z}_3^m$, defined by $\{(z_1, ..., z_m) \in \mathbf{Z}_3^m | P(z_1, ..., z_m) = 0\}$. Then, iLTS can be understood as an "intensional" representation of classical labeled transition systems, where the labels are tuples in $\mathbf{Z}_3^m$: each arrow of the iLTS labeled by $P(\bar{Z})$ intensionally represents as many arrows labeled by some $\bar{z}$ where $\bar{z} \in Sol(P(\bar{Z}))$. We call $Ext(T)$ the corresponding "extensional" labeled transition system.

Now, it is worthwhile noting that in $\mathbf{Z}_3[\bar{Z}]$, polynomials $Z_1^3 - Z_1, ..., Z_m^3 - Z_m$ evaluate to zero. Then for any $P(\bar{Z}) \in \mathbf{Z}_3[\bar{Z}]$, one for instance has $Sol(P) = Sol(P + (Z_1^3 - Z_1))$, but also, $Sol(P) = Sol(-P) = Sol(P^2)$, etc... A very natural abstraction would be to consider iLTS modulo isomorphism, of course, but also modulo $\equiv$-equivalence over labels, where $P_1 \equiv P_2$ whenever $Sol(P_1) = Sol(P_2)$[1].

[1] Also, we can consider the quotient ring $\mathbf{Z}_3(\bar{Z}) \overset{\text{def}}{=} \mathbf{Z}_3[\bar{Z}] / < Z_1^3 - Z_1, ..., Z_m^3 - Z_m >$, which is isomorphic to the ring of functions from $\mathbf{Z}_3^m$ in $\mathbf{Z}_3$ in order to restrict to polynomials of degree at most 2.

Fortunately, for algorithmic purposes, [Dut92] showed how to define a unique representative of each $\equiv$-equivalence class, called the *canonical generator*. This polynomial is the characteristic function of $Sol(P)$ and has at most degree 2.

Lemma 2. [Dut92] *Given a polynomial $P \in \mathbf{Z}_3[\bar{Z}]$, the canonical generator of $[P]_\equiv$ is computable.*

2.2 Operations over iLTS

The class of iTLS can be provided with the usual operations over (extensional) transition systems. Among them, the *parallel composition* and the *events hiding* play an important role in the complex systems design.

Parallel composition over iLTS imposes the compatibility of values between common events of the composed systems. From the extensional point of view, Definition 3 is the classical *synchronous parallel composition* as defined in ESTEREL, SIGNAL or LUSTRE languages, but the intensional approach avoids a part of the potential combinatorial explosion to compute the synchronized transitions.

Definition 3. (Parallel composition of iLTS) Let $T_1 = (Q_1, \bar{Z}, \rightarrow_1)$ be an m_1-iLTS and $T_2 = (Q_2, \bar{U}, \rightarrow_2)$ be an m_2-iLTS with possible common variables between $\bar{Z}$ and $\bar{U}$. The *parallel composition* of T_1 and T_2, written $T_1 \mid T_2$, is $(Q_1 \times Q_2, \bar{Z} \cup \bar{U}, \rightarrow)$ with

$$(q_1, q_2) \overset{P_1(\bar{Z}) \sqcap P_2(\bar{U})}{\longrightarrow} (q_1', q_2')$$

$$\text{where } P_1 \sqcap P_2 \overset{\text{def}}{=} P_1^2 + P_2^2 \text{ whenever } q_1 \overset{P_1(\bar{Z})}{\longrightarrow}_1 q_1' \text{ in } T_1 \text{ and } q_2 \overset{P_2(\bar{U})}{\longrightarrow}_2 q_2' \text{ in } T_2.$$

Because in $\mathbf{Z}_3$, $P_1 \sqcap P_2 = 0$ iff $(P_1 = 0 \wedge P_2 = 0)$, we have $Sol(P_1 \sqcap P_2) = Sol(P_1) \cap Sol(P_2)$; it entails that $(P_1 \sqcap P_2) \sqcap P_3 \equiv P_1 \sqcap (P_2 \sqcap P_3)$. Therefore, parallel composition over iLTS is commutative and associative.

Hiding events consists in abstracting from components of the label. It helps in internalizing some communications between the composed systems that are not relevant to observe in the behavior.

Let $P \in \mathbf{Z}_3[\bar{Z}]$, we shall write $\exists Z_i P$ for the polynomial $P|_{Z_i=-1} * P|_{Z_i=0} * P|_{Z_i=1}$, where $P|_{Z_i=v}$ is P obtained by instantiating any occurrence of variable Z_i by value v. The reader can check that $Sol(\exists Z_i P)$ is obtained from $Sol(P)$ by deleting the i-th component of its elements (it is a projection). Also when $\tilde{Z} \subset \bar{Z}$ is some $\{Z_{i_1}, ..., Z_{i_r}\}$ we simply write $\exists \tilde{Z} P$ for $\exists Z_{i_1} ... \exists Z_{i_r} P$.

Also it is possible to define a dual variable abstraction over polynomials, based on universal quantificator: $\forall Z_i P$ is computed as $P|_{Z_i=-1} \sqcap P|_{Z_i=0} \sqcap P|_{Z_i=1}$ which solutions are elements of the form $(z_1, ..., z_{i-1}, z_{i+1}, ..., z_m)$ s.t. $\forall z_i, (z_1, ..., z_{i-1}, z_i, z_{i+1}, ..., z_m) \in Sol(P)$. This abstraction will be considered further on.

Definition 4. (Event hiding) Let $T = (Q, \bar{Z}, \rightarrow)$ be an m-iLTS, and $Z_i \in \bar{Z}$. We define the $(m{-}1)$-iLTS $(T \setminus \{Z_i\})$ by $(Q, \bar{Z} \setminus \{Z_i\}, \rightarrow_{\setminus \{Z_i\}})$ where $q_1 \overset{\exists Z_i P}{\rightarrow}_{\setminus \{Z_i\}} q_2$ iff $q_1 \overset{P}{\rightarrow} q_2$.

2.3 Symbolic bisimulation

As we aim to manipulate transition systems in an intensional way, we explain here how the classical strong bisimulation can be handle in this setting (see Theorem 8). The definition is strongly inspired from DeSimone's symbolic bisimulation over reactive automata [DR94].
In order to be compared, events of two iLTS have to belong to the same space $\mathbf{Z}_3^m$. This way, we suppose without of lost of the generality, two iLTS have the same events variables.

Theorem 5. (Symbolic Bisimulation) *Let $T_1 = (Q_1, \bar{Z}, \rightarrow_1)$ and $T_2 = (Q_2, \bar{Z}, \rightarrow_2)$ be two iLTS. A symbolic bisimulation between T_1 and T_2 is a binary relation $\mathcal{R} \subseteq Q_1 \times Q_2$ s.t. $q_1 \mathcal{R} q_2$ whenever*

1. *for all $q_1 \overset{P}{\rightarrow}_1 q_1'$ there exists a finite set of transitions $(q_2 \overset{P_i}{\rightarrow}_2 q_2^i)_{i \in I}$ with*
   - *$(1 - P^2) * \Pi_i P_i = 0$, which implies $Sol(P) \subseteq \bigcup_i Sol(P_i)$, and*
   - *$q_1' \mathcal{R} q_2^i$, for all $i \in I$, and*
2. *vice versa.*

We have proved the congruence theorem for the symbolic bisimulation. (see appendix).

Theorem 6. (Compositionality) *Symbolic bisimulation is a congruence w.r.t. parallel composition and events hiding.*

We show here that symbolic bisimulation is an alternative view of strong bisimulation when intensional models are considered. At this stage, we assume the reader is familiar with the class of (extensional) labeled transition systems, as well as with the equivalence of *strong bisimulation*. However, we recall:

Definition 7. [Par81,Mil89b] (Strong bisimulation) Given two transition systems (labeled over some set A) $t_1 = (Q_1, A, \rightarrow_1)$ and $t_2 = (Q_2, A, \rightarrow_2)$, a bisimulation between t_1 and t_2 is a binary relation $\rho \subseteq Q_1 \times Q_2$ s.t. $(q_1, q_2) \in \rho$ whenever
(1) for all $a \in A$, for all transition $q_1 \xrightarrow{a}_1 q'_1$ there exists a state q'_2 s.t. $q_2 \xrightarrow{a}_2 q'_2$ and $(q'_1, q'_2) \in \rho$, and
(2) vice versa.

Symbolic bisimulation between iLTS corresponds to strong bisimulation between the extensional labeled transition systems:

Theorem 8. *Let T_1 and T_2 be two iLTS. Then there exists a symbolic bisimulation between T_1 and T_2 iff there exists a strong bisimulation between $Ext(T_1)$ and $Ext(T_2)$.*

2.4 Intensional Labeled Transition Systems

Intensional approach for labels offers a "compact" way to talk about sets of transitions in the system. However, we would like to reinforce this method in such a way that the whole system, and not only its sets of labels, can be itself described intensionally. For this purpose, a structure over the states is unavoidable. We propose the fairly standard structure of tuples for states where values ranges over booleans (in our setting it means values 1 and -1). This is classically used in symbolic verification methods.

Intuitively, the set of transitions will be given by a polynomial, which generalizes the iLTS approach. Applications in Section 3 will show how this formalism can be obtained for free from the real systems to be compared modulo symbolic (or equivalently, strong) bisimulation.

Definition 9. An (n, m)*-dimensional Intensional Labeled Transition System (or ILTS for short)* is a structure $S = (\bar{X}, \bar{Y}, \bar{Z}, \mathcal{T})$ where $\bar{X} = \{X_1, ..., X_n\}$ and $\bar{Y} = \{Y_1, ..., Y_n\}$ are two sets of *(source and target) states variables*, $\bar{Z} = \{Z_1, ..., Z_m\}$ is a set of labels variables and $\mathcal{T}(\bar{X}, \bar{Y}, \bar{Z})$ is a polynomial in $\mathbf{Z}_3[\bar{X}, \bar{Y}, \bar{Z}]$ describing the legal transitions.

Given some source state $\bar{x} = (x_1, ..., x_n) \in \mathbf{Z}_3^n$ and some target state $\bar{y} = (y_1, ..., y_n) \in \mathbf{Z}_3^n$, the set $Sol(\mathcal{T}(x, \bar{Z}, y))$ denotes all the possible labels of transitions from state $\bar{x}$ to state $\bar{y}$. When states are viewed extensionally, we retrieve the iLTS of in Section 2.1, which in turn can be interpreted as a classical labeled transition system.

Now, an algorithm for computing the greatest strong bisimulation between two ILTS can be described as follows. Assume given two ILTS $S_1 = (\bar{X}^1, \bar{Y}^1, \bar{Z}, \mathcal{T}_1)$ and $S_2 = (\bar{X}^2, \bar{Y}^2, \bar{Z}, \mathcal{T}_2)$.

Algorithm

1. Define the polynomial $R_0(\bar{X}^1, \bar{X}^2) = 0$.
2. Compute iteratively until stabilization the sequence of polynomials $(R_k(\bar{X}^1, \bar{X}^2))_k$
   defined by:

   $R_{k+1}(\bar{X}^1, \bar{X}^2)$ is the canonical generator of the $\equiv$-class of

$$\begin{cases} R_k(\bar{X}^1, \bar{X}^2) \\ \sqcap\ \forall \bar{Y}^1 \forall \bar{Z}[(1 - \mathcal{T}_1(\bar{X}^1, \bar{Y}^1, \bar{Z})^2) * \exists \bar{Y}^2(\mathcal{T}_2(\bar{X}^2, \bar{Y}^2, \bar{Z}) \sqcap R_k(\bar{Y}^1, \bar{Y}^2))] \\ \sqcap\ \forall \bar{Y}^2 \forall \bar{Z}[(1 - \mathcal{T}_2(\bar{X}^2, \bar{Y}^2, \bar{Z})^2) * \exists \bar{Y}^1(\mathcal{T}_1(\bar{X}^1, \bar{Y}^1, \bar{Z}) \sqcap R_k(\bar{Y}^1, \bar{Y}^2))] \end{cases} \tag{1}$$

Call $R(\bar{X}^1, \bar{X}^2)$ the result.

Theorem 10. (Termination and Correctness) *The algorithm terminates and at the end,* $R(\bar{x}^1, \bar{x}^2) = 0$ *iff there exists a bisimulation which relates states* $\bar{x_1}$ *and* $\bar{x_2}$*.*

Expression 1 can be made simpler when deterministic systems are to be compared. This is the case in Section 3, when our theory is applied to the synchronous language SIGNAL. Indeed, in this case the computation of R can be performed according to the following algorithm:

1. Compute the admissible events from a given state in each system: for system S_1, compute the canonical generator of $A_1(\bar{X}^1, \bar{Z})$ of $[\exists \bar{Y}^1 \mathcal{T}_1(\bar{X}^1, \bar{Y}^1, \bar{Z})]_\equiv$, and similarly compute $A_2(\bar{X}^2, \bar{Z})$ for S_2.
2. Compute the canonical generator $D_0(\bar{X}^1, \bar{X}^2)$ of $[\forall \bar{Z}(A_1(\bar{X}^1, \bar{Z}) - A_2(\bar{X}^2, \bar{Z}))]_\equiv$.
   Solutions of D_0 are pair of states $(\bar{x}^1, \bar{x}^2)$ which accept the same labels on their output transitions, i.e. which have the same *admissible* events.
3. Now the greatest invariant has to be computed. We iteratively compute polynomial D_k until stabilization as follows:

$$\begin{array}{l} D_{k+1}(\bar{X}^1, \bar{X}^2) \text{ is the canonical generator of the } \equiv\text{-class of} \\ \forall \bar{Y}^1 \forall \bar{Y}^2 \forall \bar{Z}[(1 - (\mathcal{T}_1(\bar{X}^1, \bar{Y}^1, \bar{Z}) \sqcap \mathcal{T}_2(\bar{X}^1, \bar{Y}^1, \bar{Z}))^2) * D_k(\bar{Y}^1, \bar{Y}^2)] \end{array} \qquad (2)$$

3 Applications

The usual synchronous programs verification practice (in particular, the verification of *safety* properties [HLR93]) needs the use of parallel composition and event hiding operations. Since the parallel composition is synchronous, the desired properties of a program can be easily and modularly expressed by means of an *observer*, i.e. another program which observes the behavior of the first one and decides whether it is correct. Then, the same formalism can be used to specify and to verify a complex system. The verification then consists in checking that the parallel composition of the two intensional transition systems never causes the observer to complain. It may happen that we just need a subset of signals: the property to verify can be expressed with this subset (for instance, the invariance under control property). It requires to specify the basic particular sets (of states and/or transitions) and to use event hiding. We need to make this handling easily available, so that program transformations remain internal and transparent, while powerful description is allowed.

As far as we are concerned, ILTS models are applied for the verification of systems described in the equational data-flow synchronous language SIGNAL [BLJ91][2]. This language is widely used to specify and to implement reactive systems as well as to verify their properties. There exists a lot

[2] developed in the EP-ATR research Group of the IRISA/INRIA Institute.

of examples using the SIGNAL environment: among them, a production cell [ALGMR95], a power transformer station controller [LBMR96], an experiment with reactive data-flow tasking in active robot vision [RMC97], ...

The original multi-clock data-flow synchronous language SIGNAL manipulates a set of *signals*; each signal $\mathtt{A}$ denotes an unbounded series of typed values $(\mathtt{A}_t)_{t\in\mathcal{T}}$, indexed by time t in a time domain $\mathcal{T}$. $\perp$ is a particular value which denotes the absence of the signal. We call *clock of* $\mathtt{A}$ the set of instants t when $\mathtt{A}$ is not absent, i.e. $\mathtt{A}_t \neq \perp$. Two signals with the same clock are called *synchronous*. The *kernel*-language SIGNAL is based on four operations, defining primitive processes by equations, and a parallel composition to combine equations, as well as a signal hiding to internalize them.

In order to simplify the presentation, we shall restrict to the boolean fragment of SIGNAL language; that is the type domain is *true*, *false* or *absent*. The constructors of the language are equations of the form $\mathtt{A} :=< \mathtt{expression} >$, as well as a parallel composition and an event hiding.

• **Static synchronous operator** $\mathtt{A} := \mathtt{p}(\mathtt{A}_1, \ldots, \mathtt{A}_\mathtt{n})$ is a boolean function of data $\mathtt{A}_1, \ldots, \mathtt{A}_\mathtt{n}$ at each instant t. This instruction requires all referred variables to have the same clock.
• **Deterministic merge operator**, written $\mathtt{A} := \mathtt{A1}\,\mathtt{default}\,\mathtt{A2}$, $\mathtt{A}$ has the value of $\mathtt{A1}$ when $\mathtt{A1}$ is present, otherwise it has the value of $\mathtt{A2}$. Its clock is the union of those of $\mathtt{A1}$ and $\mathtt{A2}$.
• **Selection operator** of the form $\mathtt{A} := \mathtt{A1}\,\mathtt{when}\,\mathtt{B}$ links $\mathtt{A}$ with $\mathtt{A1}$ when the boolean $\mathtt{B}$ has value *true*. The result can be seen as a down-sampling of a signal $\mathtt{A1}$. The clock of $\mathtt{A}$ is the intersection of that of $\mathtt{A1}$ and the set of instants when boolean $\mathtt{B}$ has value *true*.
• **Delay** (a dynamic synchronous operator) $\mathtt{A} := \mathtt{B}\,\$1$ gives access to the last value of signal $\mathtt{B}$. $\mathtt{A}$ and $\mathtt{B}$ have equal clocks. The memorizing of last values will give raise to states (see below).
• **Parallel composition** of processes is noted $|$ and consists in the conjunction of the equations (systems); it is then associative and commutative.
• **Signal hiding** $\backslash\{\mathtt{A}\}$ hides any occurrence of signal $\mathtt{A}$; it is internalized.

Logical SIGNAL programs can be translated into polynomials equations over $\mathbf{Z}_3$, following the principle of coding the possible values of a boolean signal `A` by a variable a: values for a will respectively be 1, -1 and 0 and are respectively interpreted by "`A` is *present* and *true*", "`A` is *present* and *false*", "`A` is *absent*"[3]. Therefore, any signal `A` can be associated its *clock* a^2, and two synchronous signals `A` and `B` satisfy $a^2 = b^2$.

Operators	Clock equations	Evaluations
`A` := `not A`$_1$	$a^2 = {a_1}^2$	$a = -a_1$
`A` := `A`$_1$ `and A`$_2$	$a^2 = {a_1}^2 = {a_2}^2$	$a = a_1 a_2(a_1 a_2 - a_1 - a_2 - 1)$ $a_1^2 = a_2^2$
`A` := `A`$_1$ `or A`$_2$	$a^2 = {a_1}^2 = {a_2}^2$	$a = a_1 a_2(1 - a_1 a_2 - a_1 - a_2)$ $a_1^2 = a_2^2$
`A` := `A`$_1$ `default A`$_2$	$a^2 = {a_1}^2 + (1 - {a_1}^2){a_2}^2$	$a = a_1 + (1 - {a_1}^2)a_2$
`A` := `A`$_1$ `when B`	$a^2 = {a_1}^2(-b - b^2)$	$a = a_1(-b - b^2)$
`A` := `B$1`	$a^2 = b^2$	$x' = b + (1 - b^2)x$ $a = b^2 x$

Table 1. Synchronization constraints and the boolean signal evaluation

Table 1 shows how the programs are transformed into polynomial equations (we refer to [LBBLG91] for more details), leading to an ILTS models semantics. Nevertheless, the delay operator \$ deserves some extra explanations. A delay requires to memorize the last value (then different from 0) of the signal into a (state) variable, say x. Translating `A` := `B$1`, imposes to introduce two auxiliary equations: (1) $x' = b + (1 - b^2)x$, where x' denotes the next value of state variable x, expresses the dynamics of the system. (2) $a = b^2 x$ delivers the value of the delayed signal according to the memorization in state variable x.

The translation of Table 1 is automatically performed by the SIGNAL compiler. The automata semantics can then be used as a basis for the verification of SIGNAL programs. To these ends, the tool SIGALI [DLB97], offering algebraic polynomial computations was developed. It relies on an implementation

[3] General SIGNAL programs, with other type values, can also be treated by only coding information of presence of absence of non-boolean signals.

of polynomials by Ternary Decision Diagram (TDD) (for three valued logics) in the same spirit of BDD [Bry89], but where the paths in the data structures are decorated by values in $\{-1, 0, 1\}$ instead of $\{0, 1\}$. This tool performs classic temporal logics specification verifications, whereas until now, SIGNAL programs equivalence checking was made extensionally: the tool SIGAUTO exports the TDD generated by SIGALI in order to plug other verification tools, e.g. such as ALDEBARAN [Fer84] or FCTOOLS [BRRD96]. So the result can be submitted to the tool sets for further analysis, graphical depiction, strong bisimulation, quotient computing, etc. The plug-in is achieved with the package OPEN/CAESAR.
Obviously, the size of the generated transition systems limits the extensional methods. For instance, *the transformer station on the power network* which is widely used by the French national power network is represented by a transition system with 12 state variables and 22 event variables; that is to say, this transformer station can be represented by an automaton of 2^{12} possible states and 3^{22} arrows.

The intensional methods for bisimulation checking, as proposed in this paper, perfectly fits the spirit of the tool SIGALI: the "polynomial language" provides the programmer with an intermediate language to describe algorithms over sets, in an intensional way, without bothering with the underlying implementation.
We have then improved the ILTS models semantics by implementing the algorithm of Section 2.4 for the bisimulation decision.

4 Conclusion

In this paper, we have presented *Intensional Labeled Transition Systems* intermediate models for discrete event systems. We have studied operations of parallel composition and event hiding, as well as an equivalence criterion based on strong bisimulation semantics.

The aim of this work is to rely on intensional descriptions of the systems for symbolic verification purposes, such as equivalence checking. The intensional approach we proposed has the main advantage to remain at an interesting level of abstraction in which algorithms can entirely be expressed, whereas classic symbolic approaches often suffer from a lack of algorithmic language.

Moreover, the intensional formalism is completely compatible with the symbolic technics, since intensionally described sets can be represented by standard decision diagrams.

Intensional models have already been the subject of previous work [LBBLG91], under the name of *polynomial dynamical systems*. They were the base of the temporal logics verification tool SIGALI. The results of this paper led us to enrich the scope of the verification tool SIGALI by implementing equivalence checking, such as strong bisimulation (trace equivalence, etc. are under development) on the basis of the intensional philosophy. This application is of high interest since SIGNAL is used in a lot of areas (controller synthesis [LBMR96], robotics [RMC97],...) where models equivalence checking, and on coming models reduction functionality, is crucial.

We aim now to focus on intensional approaches in its generality, in the sense that not only polynomials for finite states systems, but also other formalisms on possibly infinite systems can be investigated for the representation of sets, still remaining decidable for e.g. equivalence checking.

Acknowledgements

We are grateful to Michel Le Borgne for stimulating discussions and ideas, as well as for helping us in implemention achievement. We also would like to thank Herve Marchand for his thorough criticism of the draft.

References

[ALGMR95] T.P. Amagbegnon, P. Le Guernic, H. Marchand, and E. Rutten. Signal : The specification of a generic, verified production cell controller. *Formal Development of Reactive Systems – Case Study, Production Cell, Lecture Notes in Computer Science 891, chapitre VII*, pages 115–129, January 1995.

[BBK87] J. C. M. Baeten, J. A. Bergstra, and J. W. Klop. Decidability of bisimulation equivalence for processes generating context-free languages. In *Proc. PARLE'87, vol. II: Parallel Languages, Eindhoven, LNCS 259*, pages 94–111. Springer-Verlag, June 1987.

[BC84] G. Berry and L. Cosserat. The ESTEREL synchronous programming language and its mathematical semantics. In *Seminar on Concurrency, Pittsburgh, LNCS 197*, pages 389–448. Springer-Verlag, July 1984.

[BLJ91] A. Benveniste, P. Le Guernic, and C. Jacquemot. Synchronous programming with events and relations: the SIGNAL language and its semantics. *Science of Computer Programming*, 16:103–149, 1991.

[BRRD96] A. Bouali, A. Ressouche, V. Roy, and R. De Simone. The fc2tools set. In *Proc. of the 5th Int. Conf. AMAST'96, Munich, Germany, LNCS 1101*, July 1996.

[Bry89] R. E. Bryant. Symbolic boolean manipulation with ordered binary decision diagrams. *CM computing Surweys*, pages 293–318, September 1989.

[DLB97] B. Dutertre and M. Le Borgne. Sigali: un système de calcul formel pour la vérification de programmes signal. Technical report, Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA), July 1997.

[DR94] R. De Simone and A. Ressouche. Compositional semantics of esterel and verification by compositional reductions. *Proc. CAV'94, LNCS 818*, 1994.

[Dut92] B. Dutertre. *Spécification et preuve de systèmes dynamiques.* PhD thesis, Université de Rennes I, September 1992.

[Fer84] J.-Cl. Fernandez. *ALDEBARAN: un Système de Vérification par Réduction de Processus Communicants.* Thèse de Doctorat, Univ. Joseph Fourier-Grenoble I, France, July 1984.

[HLR93] N. Halbwachs, F. Lagnier, and P. Raymond. Synchronous observers and the verification of reactive systems. In *Proc. of the Third Int. Conf. on Algebraic Methodology and Software Technology AMAST'93, Twente, Springer Verlag*, June 1993.

[KP98] O. Kushnarenko and S. Pinchinat. Intensional approaches for symbolic methods. Technical report, Institut National de Recherche en Informatique et en Automatique (INRIA), 1998. To appear.

[LBBLG91] M. Le Borgne, A. Benveniste, and P. Le Guernic. Polynomial dynamical systems over finite fields. In G. Jacob and F. Lamnabhi-Lagarrigue, editors, *Algebraic Computing in control*, volume 165 of *Lecture Notes in Control and Information Sciences*, pages 212–222, March 1991.

[LBMR96] M. Le Borgne, H. Marchand, and E. Rutten. Formal verification of SIGNAL programs: Application to a power tranformar station controller. In *Proc. of the 5th Int. Conf. AMAST'96, Munich, Germany, LNCS 1101*, pages 270–285, July 1996.

[Mil89a] R. Milner. *Communication and Concurrency.* Prentice Hall Int., 1989.

[Mil89b] R. Milner. A complete axiomatisation for observational congruence of finite-state behaviours. *Information and Computation*, 81(2):227–247, 1989.

[Mil90] R. Milner. Operational and algebraic semantics of concurrent processes. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, vol. B*, chapter 19, pages 1201–1242. Elsevier Science Publishers, 1990.

[Par81] D. Park. Concurrency and automata on infinite sequences. In *Proc. 5th GI Conf. on Th. Comp. Sci., LNCS 104*, pages 167–183. Springer-Verlag, March 1981.

[Pla88] J. A. Plaice. *Sémantique et Compilation de LUSTRE, un Langage Déclaratif Asynchrone.* Thèse de Doctorat, I.N.P. de Grenoble, France, May 1988.

[RMC97] E. Rutten, E. Marchand, and F. Chaumette. An experiment with reactive data-flow tasking in active robot vision. *Software – Practice & Experience*, 27(5):599–621, May 1997.

Appendix

Proof of Theorem 6

Compositionality Symbolic bisimulation is a congruence w.r.t. parallel composition $|$ and events hiding $\backslash \bar{Z}$.

Proof. We only prove the result for $|$, the remaining case is easier since symbolic bisimilar states remain bisimilar when hiding is performed.

In order to simplify the notation, we write Z (resp. U) instead of $\bar{Z}$ (resp. $\bar{U}$).

Let $T_1 = (Q_1, Z, \rightarrow_1)$ and $T_2 = (Q_2, Z, \rightarrow_2)$ be two m-ILTS of dimension m, and let $T_3 = (Q_3, U, \rightarrow_3)$ be a k-ITLS.
Let assume that there exists some symbolic bisimulation $\mathcal{R}_0$ between T_1 and T_2. We show that the relation $\mathcal{R}$ given by

$$\mathcal{R} \stackrel{\text{def}}{=} \{((p_1, p_3), (p_2, p_3)) \in (Q_1 \times Q_3) \times (Q_2 \times Q_3) \mid (p_1 \mathcal{R}_0 p_2)\}$$

is a symbolic bisimulation between $T_1 \mid T_3$ and $T_2 \mid T_3$.

Consider a pair $((q_1, q_3), (q_2, q_3)) \in \mathcal{R}$. It is folklore that we have to verify the transfer property of the symbolic bisimulation.

Suppose $(q_1, q_3) \stackrel{P}{\rightarrow} (q_1', q_3')$, by Definition 3 it means that P is some $P_1(Z) \sqcap P_3(U)$ where $q_1 \stackrel{P_1(Z)}{\rightarrow}_1 q_1'$ in T_1 and $q_3 \stackrel{P_3(U)}{\rightarrow}_3 q_3'$ in T_3. By definition of $\mathcal{R}$, $q_1 \mathcal{R}_0 q_3$. Then there exists a finite set of transitions $(q_2 \stackrel{P_{2,i}(Z)}{\rightarrow} q_2^i)_{i \in I}$ with $q_1' \mathcal{R}_0 q_2^i$ and

$$[(1 - P_1^2(Z)) * \Pi_i P_{2,i}(Z)] = 0. \tag{3}$$

Now, proving that $(1 - (P_1(Z) \sqcap P_3(U)) * \Pi_i (P_{2,i}(Z) \sqcap P_3(U)) = 0$
is sufficient. Let $(z, u) \in \mathbf{Z}_3^m \times \mathbf{Z}_3^k$. If $1 - (P_1(z) \sqcap P_3(u) = 0$ then the equation is obtained; otherwise, $P_1(z) \sqcap P_3(u) = 0$, which entails $P_1(z) = 0$ and $P_3(u) = 0$. By Equation (3), $\Pi_i (P_{2,i}(z) = 0$ which implies that $P_{2,i_0}(z) = 0$ for some i_0. Therefore, $\Pi_i (P_{2,i}(z) \sqcap P_3(u)) = 0$ and we are done.
The couples (q_2^i, q_3) are then the good candidates to conclude the proof.
Transitions starting from (q_2, q_3) are dealt similarly.

Proof of Theorem 8

Let T_1 and T_2 be two ILTS. Then there exists a symbolic bisimulation between T_1 and T_2 iff there exists a strong bisimulation between $Ext(T_1)$ and $Ext(T_2)$.

Proof. $\Rightarrow$) Let $\mathcal{R}$ be a symbolic bisimulation between T_1 and T_2. We show that $\mathcal{R}$ is a strong bisimulation between $Ext(T_1)$ and $Ext(T_2)$. Let $q_1\mathcal{R}q_2$ and let $q_1 \xrightarrow{y}_1 q_1'$ in $Ext(T_1)$. Then there exists $q_1 \xrightarrow{P}_1 q_1'$ with $P(y) = 0$. By definition of $\mathcal{R}$, there exists some indexes i such that $q_2 \xrightarrow{P_i} q_2^{(i)}$, $(1 - P^2) * \Pi_i P_i = 0$ and $q_1'\mathcal{R}q_2^{(i)}$. Because $P(y) = 0$, then $(1 - P^2) * \Pi_i P_i = 0$ applied to y entails $P_i(y) = 0$ for some i which proves then $q_2 \xrightarrow{y}_2 q_2^{(i)}$, and we are done.
Transition $q_2 \xrightarrow{y}_2 q_2'$ is dealt similarly.

$\Leftarrow$) Let us show that a strong bisimulation ρ is a symbolic bisimulation. Let $q_1\rho q_2$ and let $q_1 \xrightarrow{P}_1 q_1'$ in T_1. Because $q_1\rho q_2$, for each $y_0 \in Sol(P)$ there exists $q_2 \xrightarrow{y_0}_2 q_2^{y_0}$ in $Ext(T_2)$ with $q_1'\rho q_2^{y_0}$. In T_2 there exists some P^{y_0} s.t. $q_2 \xrightarrow{P^{y_0}}_2 q_2^{y_0}$ and $P^{y_0}(y_0) = 0$.
Consider now the polynomial $\Pi_{y\in Sol(P)}P^y(Y)$[4]. Clearly, $\forall y_0,\ P(y_0) = 0 \Rightarrow \Pi_{y\in Sol(P)}P^y(y_0) = 0$, then $(1 - P^2) * \Pi_{y\in Sol(P)}P^y(Y)$ evaluates to zero. Then the q_2^y are the good candidates.

Proof of Theorem 10

In order to prove Theorem 10, we first recall some classic definitions and results (cf. [BBK87,Mil89a,Mil90]).

Definition 11. (Projective Equivalences) Let $t_1 = (Q_1, A, \rightarrow_1)$ and $t_2 = (Q_2, A, \rightarrow_2)$ be two transition systems (labeled over some set A). For $k \in \mathbf{N}$, we define the relations $\equiv_k \subseteq Q_1 \times Q_2$ by:

- $q_1 \equiv_0 q_2$ for all $(q_1, q_2) \in Q_1 \times Q_2$,
- $q_1 \equiv_{k+1} q_2$ iff
  1. for all $q_1 \xrightarrow{a} q_1'$, there exists $q_2 \xrightarrow{a} q_2'$ s.t. $q_1' \equiv_k q_2'$,
  2. reciprocally, for each $q_2 \xrightarrow{a} q_2'$, there exists $q_1 \xrightarrow{a} q_1'$ s.t. $q_1' \equiv_k q_2'$.

[4] note that $Sol(P)$ is finite.

Theorem 12. [Mil89a,Mil90] *Let $t_1 = (Q_1, A, \rightarrow_1)$ and $t_2 = (Q_2, A, \rightarrow_2)$ be two finite transition systems (labeled over some set A). For all $q_1 \in Q_1$ and $\forall q_2 \in Q_2$, we write $q_1 \underline{\leftrightarrow} q_2$ whenever there exists a bisimulation between t_1 and t_2 which relates q_1 and q_2.*
Then there exists $p \in \mathbf{N}$ s.t. $\equiv_p = \equiv_{p+1} = \equiv_{p+2} = \ldots$, and $\underline{\leftrightarrow} = \bigcap_{k=0,1,\ldots,p} \equiv_k$

Termination and Correctness The algorithm finishes and at the end, $R(\bar{x}^1, \bar{x}^2) = 0$ iff $\bar{x_1} \underline{\leftrightarrow} \bar{x_2}$.

Proof. By Theorem 12, it is enough to show that $R_k(\bar{x}^1, \bar{x}^2) = 0$ iff $\bar{x_1} \equiv_k \bar{x_2}$. We make the proof by induction over k.

1. For $k = 0$, it is obvious.
2. Suppose $R_{k+1}(\bar{x}^1, \bar{x}^2) = 0$.
   We consider a pair $(\bar{x}^1, \bar{x}^2) \in Sol(R_{k+1}(\bar{X}^1, \bar{X}^2))$. The reader can check that it is equivalent to say that:

$$
\begin{aligned}
(\bar{x}^1, \bar{x}^2) \ \in\ & Sol(R_k(\bar{X}^1, \bar{X}^2)) \\
& \bigcap\ \{(x^1, x^2) \mid \forall \bar{y}^1 \forall \bar{z} \mathcal{T}_1(\bar{x}^1, \bar{y}^1, \bar{z}) \Rightarrow \exists \bar{y}^2\ \mathcal{T}_2(\bar{x}^2, \bar{y}^2, \bar{z})\ \&\ R_k(\bar{y}^1, \bar{y}^2)\} \\
& \bigcap\ \{(x^1, x^2) \mid \forall \bar{y}^2 \forall \bar{z} \mathcal{T}_2(\bar{x}^2, \bar{y}^2, \bar{z}) \Rightarrow \exists \bar{y}^1\ \mathcal{T}_1(\bar{x}^1, \bar{y}^1, \bar{z})\ \&\ R_k(\bar{y}^1, \bar{y}^2)\}
\end{aligned}
$$

   which by induction hypothesis for R_k and Definition 11, is in turn equivalent to

$$
\begin{aligned}
(\bar{x}^1, \bar{x}^2) \ \in\ & \bar{x}^1 \equiv_k \bar{x}^2 \\
& \bigcap\ \{(x^1, x^2) \mid \forall \bar{y}^1 \forall \bar{z} \bar{x}^1 \xrightarrow{\bar{z}} \bar{y}^1 \Rightarrow \exists \bar{y}^2\ \bar{x}^2 \xrightarrow{\bar{z}} \bar{y}^2\ \&\ \bar{y}^1 \equiv_k \bar{y}^2\} \\
& \bigcap\ \{(x^1, x^2) \mid \forall \bar{y}^2 \forall \bar{z} \bar{x}^2 \xrightarrow{\bar{z}} \bar{y}^2 \Rightarrow \exists \bar{y}^1\ \bar{x}^1 \xrightarrow{\bar{z}} \bar{y}^1\ \&\ \bar{y}^1 \equiv_k \bar{y}^2\}
\end{aligned}
$$

   in other words, $\bar{x}^1 \equiv_{k+1} \bar{x}^2$.

Unit´e de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unit´e de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unit´e de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unit´e de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unit´e de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399